C8

```
%/*
%** Copyright 1997,1998 EMC Corporation
%*/

/*
**  Leading % causes rpcgen to pass a line directly thought to the output,
**  ie edmlink_sunrpc.h in this case.  This allows the .h to make a little
**  more sense and be properly documented.
*/

%/*
% * dispatch_daemon.x : EDM Dispatch Daemon C/S communication module
% *
% * Mission Statement:  This is an RPCGEN file which defines the RPC interface
% *                     between the Dispatch Daemon (which resides on
% *                     the EDM server) and  the backup client callers of its
% *                     functions.  This defines the RPC level calls that a
% *                     "caller" can make and the "service" will respond to.
% *
% * Primary Data Acted On: This defines the data that will flow over the wire.
% *                        The RPC mechanism will take care of data
                                                                 marshalling
% *
% *
% * Compile-Time Options:
% *                     This acutally gets run through RPCGEN not compiled.
                                                                          It
% *                      must be run through with the -h flag to create a
% *                      header, the -m flag to create the service side
% *                      routines, the -l flag to create the client side
% *                      routines, and the -c flag to create the common data
% *                      marshalling routines.
% *
% * Basic idea here:
% *                     Define the RPC level interfaces to the Dispatch Daemon
% *                     and all data types that will be passed via RPC.
%*/


/**********************************************************************
 Constant Definitions
**********************************************************************/


/**********************************************************************
 Data Structure Definitions
**********************************************************************/


        struct DD_rpc_objID
        {
          int           type;    /* Object identifier (DD_OTYPE_*) */
%#define DD_OTYPE_INIT_IN        1     /* Initialize Input Object */
%#define DD_OTYPE_INIT_OUT       2     /* Initialize Output Object */
          long          len;     /* Length of structure, version number */
        };


        struct DD_client_session_id {
                unsigned long   high;
                unsigned long   low;
        };

const DD_SERVICE_RESTORE=1;
/* structures for input and output of re_initialize rpc call:  */
        struct DD_initialize_args {
                int      service;
```

```
                string hostname<>;
                string username<>;
                unsigned int timeout;
        };

const DD_SERVICE_FAILURE_NONEXEC=-4;
const DD_SERVICE_FAILURE_PERMS=-2;
const DD_SERVICE_FAILURE_EXEC=-1;
const DD_SERVICE_STARTING=1;
const DD_SERVICE_RUNNING=2;
const DD_SERVICE_COMPLETED=4;

        struct DD_initialize_result {
                unsigned int status;
                DD_client_session_id service_handle;
        };

/* structures for getstatus function */
        struct DD_getservicestatus_args {
                int             status;
                DD_client_session_id    service_handle;
        };

        struct DD_getservicestatus_result {
                int             status;
                opaque          handle<128>;
        };

/* work item type */
/*
 * These match the rbconfig.h for the most part. There are
 * some extras for identifying NOS workitems.
 */
const FS_BACKUP_TYPE              =       0;
const SHARED_PART_BACKUP_TYPE     =       1;
const SHARED_M_PART_BACKUP_TYPE =       2;
const OFFLINE_DB_TYPE             =       3;
const ONLINE_KICKDB_TYPE          =       4;
const ONLINE_LISTDB_TYPE          =       5;
const DCONN_KICK_TYPE             =       6;
const DCONN_NET_TYPE              =       7;
const DCONN_WRK_TYPE              =       8;

/* length of various buffers */
const MEDNAME_SIZE =    6;
const TRLNAME_SIZE =    16;
const WINAME_SIZE =     64;
const TEMPLNAME_SIZE =  64;
const USERNAME_SIZE =   64;
const HOSTNAME_SIZE =   256;
const CLNTNAME_SIZE =   64;
const SERVER_SIZE =     256;
const MAX_STRING_SIZE = 256;  /* must be the length of the longest buffer */

/* defines for operation_type */
const BACKUP_TYPE =      1;
const RESTORE_TYPE =     2;
const OTHER_TYPE =       16;

/* work item structure */
struct WIProgress {
    unsigned long       time_started;
    unsigned long       curr_time;

    unsigned long       total_kbytes_sofar;
    unsigned long       total_files;
    unsigned long       total_badfiles;

    unsigned long       curr_kbytes_sofar;
    unsigned long       curr_time_slice;
    unsigned long       curr_files;

    unsigned long       total_files_expected;
    unsigned long       total_kb_expected;

    int                 operation_type;
    int                 completed;
    unsigned long       status;

    struct WIProgress     *next;

    char                wi_name[WINAME_SIZE];
    char                trail_name[TRLNAME_SIZE];
    char                trailset_name[TRLNAME_SIZE];
    char                template_name[TEMPLNAME_SIZE];
    char                client_name[CLNTNAME_SIZE];
    char                server_name[SERVER_SIZE];
    char                media_type[MEDNAME_SIZE];
    char                userid[USERNAME_SIZE];

    char                level;
    char                type;
};

/* SUMMARY structure */
struct EDMProgress {
    unsigned long       time_started;
    unsigned long       curr_time;

    unsigned long       total_kbytes_sofar;
    unsigned long       total_files;
    unsigned long       total_badfiles;

    unsigned long       curr_time_slice;
    unsigned long       curr_kbytes_sofar;
    unsigned long       curr_files;

    unsigned long       active;
    unsigned long       total;
    unsigned long       failed;
    unsigned long       successful;

    unsigned long       total_files_expected;
    unsigned long       total_kb_expected;

    int                 operation_type;
    int                 completed;
    unsigned long       status;

    struct EDMProgress    *next;

    char                host_name[HOSTNAME_SIZE];
};

struct EDMStats
{
    unsigned long        status;
    EDMProgress          edm;
    WIProgress           *wiprogress;
};

struct CC_Notify
```

```
{
    int                         msgtype;
    int                         sourcemodule;
    int                         level;
    int                         msglen;
    string                      msgtext<>;
};

struct SessionInfo
{
        DD_client_session_id    service_handle;
        unsigned int            status;
        unsigned long           jobstarttime;
        int                     operation_type;
        long                    lastSent;
        long                    lastReceived;
        int                     outhandle;
        int                     errhandle;

       SessionInfo              *next;
};

struct SessionBlock
{
        struct SessionInfo      *sess;
        int                     totalsessions;
};

program EDM_DISPATCH_DAEMON {

   version EDMDD_FUNCTIONS {

            /* Functions for EDMRST_Initialize */
            DD_initialize_result dd_initialize( DD_initialize_args ) = 1;

            DD_getservicestatus_result dd_getservicestatus(
                                        DD_getservicestatus_args ) = 2;

            SessionBlock dd_getsessioninfo( DD_getservicestatus_args ) = 3;

   } = 1; /* This is version 1 */

%/* This is the RPC program number.
                                       These are reserved in /pds/docs/RPC_numbers
% * This number cannot be re-used by any other RPC daemon on the machine,
                                                                          as it
% * identifies this daemon uniquely.  If it were to be re-used,
                                                                the last daemon
% * to register would be contacted when RPC's come in for this number.
%*/
} = 390015;
```

```
%/*
%** Copyright 1997,1998 EMC Corporation
%*/

/*
**  Leading % causes rpcgen to pass a line directly thought to the output,
**  ie edmlink_sunrpc.h in this case.  This allows the .h to make a little
**  more sense and be properly documented.
*/

%/*
% * dispatch_daemon.x : EDM Dispatch Daemon C/S communication module
% *
% * Mission Statement:  This is an RPCGEN file which defines the RPC interface
% *                     between the Dispatch Daemon (which resides on
% *                     the EDM server) and  the backup client callers of its
% *                     functions.  This defines the RPC level calls that a
% *                     "caller" can make and the "service" will respond to.
% *
% * Primary Data Acted On: This defines the data that will flow over the wire.
% *                        The RPC mechanism will take care of data
                                                                  marshalling
% *
% *
% * Compile-Time Options:
% *                     This acutally gets run through RPCGEN not compiled.
                                                                           It
% *                      must be run through with the -h flag to create a
% *                      header, the -m flag to create the service side
% *                      routines, the -l flag to create the client side
% *                      routines, and the -c flag to create the common data
% *                      marshalling routines.
% *
% * Basic idea here:
% *                     Define the RPC level interfaces to the Dispatch Daemon
% *                     and all data types that will be passed via RPC.
%*/

%#include <restore/dispatch_daemon.h>

/*********************************************************************
 Constant Definitions
*********************************************************************/
#define DMA_RESTORE_SERVICE "RESTORE_SERVICE"
const   MAX_STRING_SIZE = 256;
#define  TEXT  1
#define LONG  2
#define INT   3
#define CHAR  4

/*********************************************************************
 Data Structure Definitions
*********************************************************************/

/* work item type */
/*
 * These match the rbconfig.h for the most part. There are
 * some extras for identifying NOS workitems.
 */
const FS_BACKUP_TYPE              =       0;
const SHARED_PART_BACKUP_TYPE     =       1;
const SHARED_M_PART_BACKUP_TYPE =         2;
const OFFLINE_DB_TYPE             =       3;
const ONLINE_KICKDB_TYPE          =       4;
const ONLINE_LISTDB_TYPE          =       5;
const DCONN_KICK_TYPE             =       6;
```

```
const DCONN_NET_TYPE              =       7;
const DCONN_WRK_TYPE              =       8;

/* length of various buffers */
const MEDNAME_SIZE =    6;
const TRLNAME_SIZE =    16;
const WINAME_SIZE =     64;
const TEMPLNAME_SIZE =  64;
const USERNAME_SIZE =   64;
const HOSTNAME_SIZE =   256;
const CLNTNAME_SIZE =   64;
const SERVER_SIZE =     256;
const MAX_STRING_SIZE = 256;  /* must be the length of the longest buffer */

/* defines for operation_type */
const BACKUP_TYPE =      1;
const RESTORE_TYPE =     2;
const OTHER_TYPE =       16;

/*********************************************************************
             M E S S A G E   D E F I N I T I O N S
*********************************************************************/
struct DP_connect_indicate_msg {
        DD_client_session_id sid;
};

struct DP_connect_confirm_msg {
        DD_client_session_id sid;
};

struct DP_abort_request_msg {
        DD_client_session_id sid;
};

struct DP_abort_response_msg {
        DD_client_session_id sid;
        uint32  terminationCode;
};

struct DP_close_request_msg {
        DD_client_session_id sid;
};

struct DP_close_response_msg {
        DD_client_session_id sid;
        uint32  terminationCode;
};

struct DP_ping_request_msg {
        DD_client_session_id sid;
};

struct DP_ping_response_msg {
        DD_client_session_id sid;
};

union dataparms switch( uint32 data_type ) {

        case TEXT:
                char string_data[ MAX_STRING_SIZE ];
        case CHAR:
                char char_data;
        case LONG:
                unsigned long ulong_data;
        case INT :
                int integer_data;
        default  :
                void;
};

struct DP_event_indicate_msg {
        DD_client_session_id sid;
        uint32          EventNumber;
        string          EventText<>;
        uint32          EventLevel;
        dataparms       parms;
};

struct DP_event_confirm_msg {
        DD_client_session_id sid;
};

struct DP_progress_indicate_msg {
        DD_client_session_id sid;
        EDMStats stats;
};

struct DP_progress_confirm_msg {
        DD_client_session_id sid;
        uint32  ack;
};

struct DP_final_stats_indicate_msg {
        DD_client_session_id sid;
        EDMStats stats;
};

struct DP_final_stats_confirm_msg {
        DD_client_session_id sid;
        uint32  ack;
};
```

```
%/*
%** Copyright 1997,1998 EMC Corporation
%*/

/*
**  Leading % causes rpcgen to pass a line directly thought to the output,
**  ie edmlink_sunrpc.h in this case.  This allows the .h to make a little
**  more sense and be properly documented.
*/

%/*
% * dispatch_daemon.x : EDM Dispatch Daemon C/S communication module
% *
% * Mission Statement:  This is an RPCGEN file which defines the RPC interface
% *                     between the Dispatch Daemon (which resides on
% *                     the EDM server) and  the backup client callers of its
% *                     functions.  This defines the RPC level calls that a
% *                     "caller" can make and the "service" will respond to.
% *
% * Primary Data Acted On: This defines the data that will flow over the wire.
% *                        The RPC mechanism will take care of data
                                                                 marshalling
% *
% *
% * Compile-Time Options:
% *                     This acutally gets run through RPCGEN not compiled.
                                                                          It
% *                      must be run through with the -h flag to create a
% *                      header, the -m flag to create the service side
% *                      routines, the -l flag to create the client side
% *                      routines, and the -c flag to create the common data
% *                      marshalling routines.
% *
% * Basic idea here:
% *                     Define the RPC level interfaces to the Dispatch Daemon
% *                     and all data types that will be passed via RPC.
%*/

%#include <restore/dispatch_protocol.h>

/********************************************************************
 Constant Definitions
********************************************************************/


/********************************************************************
 Data Structure Definitions
********************************************************************/

program EDM_DISPATCH_PROTOCOL_CLIENT {

   version EDMDPC_FUNCTIONS {

     int dp_connect_confirm( DP_connect_confirm_msg ) = 1;
     int dp_abort_request( DP_abort_request_msg ) = 2;
     int dp_close_request( DP_close_request_msg ) = 3;
     int dp_ping_request( DP_ping_request_msg ) = 4;
     int dp_event_confirm( DP_event_confirm_msg ) = 5;
     int dp_progress_confirm( DP_progress_confirm_msg ) = 6;
     int dp_final_stats_confirm( DP_final_stats_confirm_msg ) = 7;

   } = 1; /* This is version 1 */

%/* This is the RPC program number.
                                        These are reserved in /pds/docs/RPC_numbers
% * This number cannot be re-used by any other RPC daemon on the machine,
```

```
                                                                       as it
% * identifies this daemon uniquely.  If it were to be re-used,
                                                              the last daemon
% * to register would be contacted when RPC's come in for this number.
%*/
} = 399999;
```

```
%/*
%** Copyright 1997,1998 EMC Corporation
%*/

/*
**  Leading % causes rpcgen to pass a line directly thought to the output,
**  ie edmlink_sunrpc.h in this case.  This allows the .h to make a little
**  more sense and be properly documented.
*/

%/*
% * dispatch_daemon.x : EDM Dispatch Daemon C/S communication module
% *
% * Mission Statement:  This is an RPCGEN file which defines the RPC interface
% *                     between the Dispatch Daemon (which resides on
% *                     the EDM server) and  the backup client callers of its
% *                     functions.  This defines the RPC level calls that a
% *                     "caller" can make and the "service" will respond to.
% *
% * Primary Data Acted On: This defines the data that will flow over the wire.
% *                          The RPC mechanism will take care of data
                                                                 marshalling
% *
% *
% * Compile-Time Options:
% *                     This acutally gets run through RPCGEN not compiled.
                                                                          It
% *                      must be run through with the -h flag to create a
% *                      header, the -m flag to create the service side
% *                      routines, the -l flag to create the client side
% *                      routines, and the -c flag to create the common data
% *                      marshalling routines.
% *
% * Basic idea here:
% *                     Define the RPC level interfaces to the Dispatch Daemon
% *                     and all data types that will be passed via RPC.
%*/

%#include <restore/dispatch_protocol.h>

/********************************************************************
 Constant Definitions
********************************************************************/


/********************************************************************
 Data Structure Definitions
********************************************************************/

program EDM_DISPATCH_PROTOCOL_SERVICE {

   version EDMDPS_FUNCTIONS {

      int dp_connect_indicate( DP_connect_indicate_msg ) = 1;
      int dp_abort_response( DP_abort_response_msg ) = 2;
      int dp_close_response( DP_close_response_msg ) = 3;
      int dp_ping_response( DP_ping_response_msg ) = 4;
      int dp_event_indicate( DP_event_indicate_msg ) = 5;
      int dp_progress_indicate( DP_progress_indicate_msg) = 6;
      int dp_final_stats_indicate( DP_final_stats_indicate_msg ) = 7;

   } = 1; /* This is version 1 */

} = 399998;
```